RECORD OF ORAL HEARING

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex Parte DAVID ROBERT BALDWIN

Appeal 2009-003677
Application 09/591,225
Technology Center 2600

Oral Hearing Held: August 11, 2009

Before MASHID D. SAADAT, KARL D. EASTHOM, and ELENI MANTIS MERCADER, *Administrative Patent Judges.*

ON BEHALF OF THE APPELLANT:

Robert O. Groover, III, Esquire
CREATIVE/3DLABS/GROOVER
P.O. Box 802889
Dallas, TX 75380

The above-entitled matter came on for hearing on Tuesday, August 11, 2009, commencing at 9:00 a.m., at The U.S. Patent and Trademark Office, 600 Dulany Street, Alexandria, Virginia, before Victor Lindsay, Notary Public.

MS. BOBO-ALLEN: Calendar Number 31, Appeal Number 2009-3677, Mr. Groover.

JUDGE SAADAT: Thank you.

Are you Mr. Groover?

MR. GROOVER: Yes, I am.

JUDGE SAADAT: And we have read the record and you can start with your main point.

MR. GROOVER: Great. I'm sorry, say the last part of it again.

JUDGE SAADAT: You can start with your main point. We are familiar with the case and --

MR. GROOVER: Excellent.

JUDGE SAADAT: -- the Brief and everything.

MR. GROOVER: Let me, if I may, just preface with one brief note about the history of this case. This case has had six nonfinal Office actions over six different combinations of references with the same claims and with no RCEs. So we're still, I'm going to suggest -- don't have a reference which is very close. I would ask that if any of Your Honors wish to indicate a new reference to be applied and remand, which sometimes happens, please suggest special status. This case has been a long time getting to the Board.

JUDGE SAADAT: And the same references have been applied six times?

MR. GROOVER: No, no, six different sets of references have been applied.

JUDGE SAADAT: Oh, okay.

MR. GROOVER: And so this is now the eighth Office action that we're appealing from and, indeed, in this case there were new grounds of -- well, new references brought up in the advisory action, which also I have never seen in my 30 years. So perhaps Your Honors have seen more extraordinary prosecutions, but I don't believe that I have.

I would suggest, Your Honors, the main point in this case is very simple. If we look at Figure 2 of the European reference supplied, the Kaiser reference, it's a short simple patent application, and what it shows is that it's about synchronizing the address translation and about the requirements of providing a copy of the appropriate page translation map so that you can have a local TLB for use by the graphics engine.

There has been some really puzzling argument, at least puzzling to me, that use of a translation lookaside buffer is the same as handling a page miss. That's not how I've understood virtual memory management for -- at least for my 25 years or so exposure to it, and we've pounded that point to death, I think. Do Your Honors have any questions about it? I'd be happy to discuss --

JUDGE EASTHOM: I think that's the central issue. You know, I guess the Examiner's saying that this Nachos reference kind of indicates a TLB page miss as somehow a page fault. And I think your point is that Nachos, the whole Nachos is -- involves a host processor. Is that --

MR. GROOVER: Nachos appears -- yes, Your Honor. Nachos appears to be a fairly startling development where they are using only a TLB, a fully associative memory, instead of an oral page table. So Nachos may be an old publication or a very specialist architecture.

Now in that case, if there is no page table, then, yes, it is possible to make an architecture where a TLB miss automatically implies a page fault. But by the same analogy, one might say that just because automobiles sometimes have wrecks, that does not imply that an automobile is a wreck, unless, of course, my daughter's driving it, so --

JUDGE EASTHOM: Maybe you can back up a little bit and --

MR. GROOVER: Please.

JUDGE EASTHOM: -- just explain exactly what you see as a page fault.

MR. GROOVER: Yes. Yes, Your Honor.

JUDGE EASTHOM: I know it would help us a little bit.

MR. GROOVER: So virtual memory -- and there's a lot of background context discussed in our own lengthy application and in the other even lengthier applications which it incorporates. In virtual memory management we're translating a virtual address to a physical address and the translation changes. So one of the basic types of caching -- so there's sort of two areas of memory management. One is cache and one is virtual memory management. TLB is simply a particular type of caching which is used to perform the virtual address translation rapidly. In fact, I glanced on Wiki this morning to make sure that I wasn't totally off in space, and they commented, yes, there's three basic types of caching, data instruction and TLB, and the Kaiser reference shows that quite clearly. So if virtual address such and such at the present corresponds to a certain physical address, Kaiser is related to the necessity for the graphics processor to be able to find out that table and quickly access it. In other words, Kaiser passes that work

off to the -- sorry, passes the tools to do the translation to the graphics processor so as not to require the host to pin down those correspondences during the whole time when the graphics processor is running a particular task.

Shall I go on more or is that --

JUDGE EASTHOM: I was -- just on a simpler level --

MR. GROOVER: Please.

JUDGE EASTHOM: -- the way I understand it, you have what you call disk memory or host memory, right? And that's a separate part from the -- if you have --

MR. GROOVER: Right.

JUDGE EASTHOM: So TLB doesn't recognize the address, the prior art would have to -- and it can't find -- I guess, if there's a TLB miss, even in Nachos, the virtual address won't match up to a regular address that points to the memory within --

MR. GROOVER: Those are two different steps, Your Honor.

JUDGE EASTHOM: Okay.

MR. GROOVER: Let's slow down a little bit and look at what the virtual memory system itself does. If you look at -- let's see, into the drawings of our application, which I apologize are rather ugly. Well, Figure 1 is a, it appears a computer diagram, but let's refer to it for just a second. So we have a CPU, which in this one is at the -- good Lord, where is it? Right, 425.

JUDGE EASTHOM: Let's start --

MR. GROOVER: Yes, sir?

JUDGE EASTHOM: I mean, in your spec on page 8, you kind of say that -- and I know you and the Examiner talked about what this says, but it says "In particular, the present application discloses a computer system in which a graphics accelerator unit manages page faulting of texture data invisibly to the host processor. When a logical page fault occurs and the page of texture is in the second level of memory, i.e. the host's physical memory, it will be fetched in automatically by the graphics memory manager." So I take it to mean that a logical page fault means that this texture data that you're trying to find is in -- over by the host's memory. Is that incorrect?

MR. GROOVER: Yes. Yes, Your Honor. There's two places -- well, sometimes. There are two places where the texture memory can reside, and our application goes into that. The texture data may have already been downloaded into the graphics accelerator, in which case it's in the local specialized memory, or it may still be sitting out in the host's physical memory. Now when we refer to host's physical memory, that does not include going to disk. Going to disk is still a third stage.

JUDGE EASTHOM: Oh, okay.

MR. GROOVER: That is referred to as bulk memory. So in other words, we will have dedicated memory on the graphics card, which these days might be a gig or so. We will have the host's own semiconductor memory, which these days might be, depending on how much you pay, might be more gigs. And then we will have the disk itself, which these days might be a terabyte or so. So typically, when you have an active -- and it's sort of a like a hierarchy of caching in that you want to have the most

frequently accessed data closest to the graphics processor, because, of course, part of what drives this whole area is that graphics processing is extremely data transfer intensive. So in a graphics card -- and I'm sorry we don't have a drawing, I think, which -- if you look at our Figure 3, this would show an example of a graphics card. Yes. So we have some local memory on the left side. We have a core, which is doing various applications very fast. It has got interfaces to the, you know, general purpose bus as well as the dedicated memory bus, as well as other dedicated memory buses. So in that graphics card, you hope that as much as possible of the fast data shuffling is going to happen with the local memory. So when you hit your -- so, for instance, when your graphics processor starts a task and hits the translation lookaside buffer, that should contain data which will let it find out whether the current requested piece is local or not. If it isn't local -- and here comes part of the difference from the prior art. If it isn't local, then our implementation allows the graphics processor to go out and sneak under the host processor's nose and pull in what it needs.

JUDGE EASTHOM: Well, if it isn't local, that's a page fault?

MR. GROOVER: Yes. If it's not loaded into the current working set, that's a page fault. So for instance, your graphics processor will be happily chugging along using virtual addresses and finding some of that data in its local memory. But if it's not in the local memory, then we have to go out to main memory, which slows things down, partly because the graphics processor doesn't have first priority on the main memory and, you know, bus latency and other such. So that would be a page fault.

Now let's back up and look at Kaiser. Kaiser, you can see from the title, you can see from the claims, you can see from --

JUDGE EASTHOM: Well, I know Kaiser -- I don't want to belabor the issue. I know Kaiser, specifically, you know, defined a page fault and it's different from a TLB miss, and --

MR. GROOVER: Yeah.

JUDGE EASTHOM: -- a page fault, he is interrupting the processors.

MR. GROOVER: Right.

JUDGE EASTHOM: I understand that argument.

MR. GROOVER: Right. Right.

JUDGE EASTHOM: I think the Examiner's argument is that this TLB miss -- I don't really totally understand his argument, to be honest with you. It seems that he's saying that if there is a TLB miss, that this graphics processor in Kaiser is going to go look elsewhere for it. That's the crux of the issue, I think, isn't it?

MR. GROOVER: Your Honor, Kaiser doesn't even show the data path. All Kaiser is showing is the address translation path. If you look -- in fact, if you look at Kaiser --

JUDGE EASTHOM: In other words -- he's showing the translation path. So I guess you're saying that the path is going to be just looking for data that's local no matter what, if it's a TLB miss?

MR. GROOVER: Even narrower than that. Kaiser is about -- if you look at the bottom left of Kaiser's Figure 2. We have that local TLB, but we also have a TLB sync input because we have to keep the page mappings in this buffer 226 synchronized with the main page table that's out in system

memory. So when it shows a connection into memory 24, what it's specifically talking about is to synchronize the page tables. If you look at the last half column of Kaiser, it talks about there's some additional mechanism required, and then Kaiser goes into how the R bits and C bits are used. But all of that, all of that, as far as I can tell, is purely related to getting an effective address.

Now, what happens with the effective address, that's not shown. That's not on this diagram at all. The only connection to main memory we see is from the page table buffer and -- well, I don't know what that additional input is. Kaiser's a very short publication and it specifically references all the other power PC documentation. You know, that's a huge volume. And if there is something in that huge volume that said, that anticipated our invention, I'd like to see it. But this is not showing -- in fact, it's specifically not showing full memory management on the graphics engine. In other words, the host defines the translation table and allows the graphics engine to keep a local copy of it, as best I can tell.

Am I answering that, Your Honor, or --

JUDGE EASTHOM: So the -- I thought the Examiner's -- so your position is, is that even the TLB is managed by the host? I didn't really pick up that argument before from you.

MR. GROOVER: The TLB -- if you look at how it talks about when there is a TLB miss, and we go into table walk logic -- that's conventional. That's how you would normally update a TLB in a virtual memory system. That goes to the page table buffer. Well, what the page table buffer for is for synchronizing the page table as between the host and the card.

JUDGE EASTHOM: The page table, is that a page of data or is that the page of addresses?

MR. GROOVER: Perfect question. The page table is the master map of translation from virtual to physical. So the page table is your basic map to do this translation. The TLB is just a local cached copy. So anytime you have to do a table look up, you're losing cycles. But if you -- the TLB is to do the ones that are local as fast as possible.

JUDGE EASTHOM: So the TLB is also an address exchanger, if you will?

MR. GROOVER: Yes.

JUDGE EASTHOM: But it has to access the page table buffer to --

MR. GROOVER: Well, it has to be updated. So in other words, the relation between -- and this is what I meant about caching. The relation between the TLB and the page table is a caching relation. The TLB is just keeping the most recently used part of your page table close to the processor so you can make the translations you need as fast as possible.

JUDGE EASTHOM: Okay. And, now, if there's a TLB miss, would this graphics processor -- what happens then? What happens --

MR. GROOVER: If there's a TLB miss, then you have to update your TLB. So you would use -- in this example, your table walk logic would update that from the page table buffer. But all this is just about updating the page table and the more local version of it.

JUDGE EASTHOM: And then what would happen? You would update it and then go into local memory to find the data you need or --

MR. GROOVER: Well, there's a couple of ways that could happen from that. Probably, if you have to update it, it's not in local memory. This is the point where Kaiser is, as far as I can tell, is at the point of further deponents sayeth not.

JUDGE EASTHOM: Say that again. I'm sorry.

MR. GROOVER: This is the point where Kaiser is, to make a lawyer joke, is at the point of further deponents sayeth not.

JUDGE EASTHOM: Oh, I got you.

MR. GROOVER: In other words, once there is a correct translation there, now your graphics engine is going to have a correct physical address, which might be a local physical address or it might be it has to make a DMA hit and get a block of memory loaded into -- you know, there's some vague language about that. That part is mostly not what Kaiser is about. So I'm guessing that either it is according to the -- you know, either a conventional access or something according to the power PC system, and I think we would have heard about it if this were in power PC.

So, you know, could there be a better reference than this? I hope so. If Your Honor -- you know, we don't know of one. If Your Honors learn of a better reference, we'd love to know about it. But this has been a long and frustrating prosecution and we're not still at all close to the invention that I can see.

Let me -- sorry?

JUDGE SAADAT: I don't have a question.

MR. GROOVER: Let me mention one thing, Your Honors. I have asked myself, you know, why has this particular case gone through eight

Office actions without any RCE. If it's not my winning personality, then I have guessed that it might be that Claim 1. You know, Claim 1 is an extremely short claim. I think that we're entitled to it, but, you know, some Examiners -- well, you very seldom see a two-line claim in the predictable arts. So let me offer at this point, if this would simplify issues, we would be happy to cancel Claim 1 right now without prejudice. All of the other claims, you can see, are much more detailed and hardware-ish. Would that perhaps simplify consideration? No promises, understand.

JUDGE SAADAT: Well, we just consider --

JUDGE EASTHOM: What did you say? We deponents not --

JUDGE SAADAT: We can't cancel claims. We can't allow claims.

MR. GROOVER: Oh, I'm sorry?

JUDGE SAADAT: The Board doesn't have the authority to cancel claims or allow any claim.

MR. GROOVER: I believe that I can cancel that claim at oral argument. I have done this before and I have been reassured by Board judges that it is possible. If this would simplify the case, I would be -- you know, part of what I've been asking myself is somebody has strong feelings on this case and perhaps it's because I have got such an extremely short claim. And, you know, I don't need to be bravure about this. So if it would -- unless you're going to tell me stop, no, no, then I'm offering hereby to cancel Claim 1 and its dependents right now without prejudice.

JUDGE MERCADER: Counsel, can you tell us, in Claim 1 we noticed wordings that it didn't appear in other claims even though Claim 1 is pretty short, the invisibly part.

MR. GROOVER: Right.

JUDGE MERCADER: So why -- is that an important feature, and if that's important, why isn't it in the other independent claims?

MR. GROOVER: It is a -- that feature does actually appear in some other claims, Your Honor, and let me -- we have quite a few independent claims in this case. Right. Some of the other claims use a key word of automatically.

JUDGE MERCADER: But is automatically the same thing as invisibly? I don't think so, correct?

MR. GROOVER: I will happily keep the -- right, Claim 3, for instance, specifically says invisibly, and --

JUDGE MERCADER: Claim 3 does?

MR. GROOVER: Yes. Right at the very last line. As does Claim 14.

JUDGE EASTHOM: I think 15 does. I had a list. I think some of your claims have page faulting and invisibly, and the independent claims that don't have page faulting also don't have invisibly, I don't think.

MR. GROOVER: Okay. So, you know, page faulting is an established term. It's been known for decades. There's not any -- you know, not that I've ever seen, any serious technical dispute about what that term means. You know, this stuff comes at least from the IBM 370 work which was in the early '70s, I think. So I would respectfully submit that the language is clear enough.

And also, remember, that after eight Office actions, you know, we've been sand-bagged with this new reference in the advisory action. I've never seen a new reference in an advisory action.

JUDGE EASTHOM: Well, I -- with all due respect to you, I think he's just trying to, you know, determine what the -- he's using it as an evidentiary reference to maybe figure out how broad he can stretch the word page faulting.

MR. GROOVER: Yes, Examiner. But as -- yes, Your Honor. But --

JUDGE EASTHOM: That's all right. I'm used to that.

MR. GROOVER: But if the Examiner brings in something at such a late stage, we can't adduce evidence to reply to it. That's very important. If there's, you know, real contest about that, you know, we can bring in expert witnesses. We can bring in copies of books. Some of the books that are referred to in, I think, you know, that same Nachos things are, you know, books I've got on my shelf and which I incorporated reference in the -- into this application, so it's not -- you know, this is not mysterious technology. Because the chip manufacturers have to put out datasheets, there's not much secret technology in this particular art.

JUDGE EASTHOM: Well, I wasn't real -- I wasn't -- I didn't think the Examiner -- I mean, after reading both the arguments, it seemed to me that your argument is that even in Nachos when they refer to page fault, the host processor's involved. Is that -- am I not --

MR. GROOVER: Nachos is an odd one because the TLB is smashed together. In other words, there apparently no page table at all, which is puzzling. Does the host processor have anything to do with that? I don't know Nachos well enough to make any definite assurances.

JUDGE EASTHOM: What does -- when it refers to the operating system in Nachos, are you -- what does -- what are they talking about there?

MR. GROOVER: Well, so Nachos talks about, I think, if there's a page fault twice, then you would have to branch to an operating system. That's basically, as I understand it at least. That's a normal provision where if you have a stuck-at fault -- in other words, if your process is trying to access a certain memory over and over and over again, then the OS is going to need to respond to that with some error response, which will range into a higher level error handling routine. So I don't think that relates to anything like what we're talking about.

JUDGE EASTHOM: So, specifically, when the Examiner says that Nachos shows that a TLB miss can be a page fault, what is your response to that? I thought --

MR. GROOVER: The simple answer is that cars can have wrecks, but a car is not the same as a wreck. You know, Nachos has combined two things and -- surprisingly. And, therefore, if you have a TLB miss, because there's nothing else, that in Nachos' particular context means a page fault. But that's not normally true. And the concepts are very different. Our reply brief, I think, goes into some pretty good detail on that.

JUDGE EASTHOM: I thought the point of your reply brief was that Nachos involved a processor with a TLB, but maybe I missed the point.

MR. GROOVER: The point of our response to Nachos, if I can paraphrase it, is that Nachos appears not to have a page table. So, normally, the TLB -- and, in fact, you might say it the other way. Although Nachos doesn't use these words, you could say that Nachos doesn't have a TLB. In other words, normally, the TLB is a cache of the page table. But in Nachos, that doesn't happen. They're smashed into the same thing. And, therefore,

normally you go through your TLB, you go in your page table, and then if it's not there, you have a page fault. But because in Nachos there's no separation there, a miss becomes a page fault.

JUDGE EASTHOM: Oh, I understand.

MR. GROOVER: Okay. So that's -- that is simply because of the peculiarity in Nachos. And, again, if the Examiner is trying to suggest that a TLB miss is the same as a page fault, well, you know, we can respond to that, you know, but we were never given a chance. And we're not allowed to put any evidence with our Reply Brief. We tried to be pretty definite in our arguments, but, you know, we're not allowed to put in any evidence for this, you know, last minute assertion.

May I ask, Your Honors, before I run out of time, is there any question or doubt about the question of two memory management units? In other words, that there is virtual memory management going on on the host and, according to Claim 3 and that whole family of claims, we also have a memory manager -- the Examiner misquoted our language, by the way, but we do have a second memory management unit in the localized graphics card, which can do its own independent operations.

JUDGE MERCADER: Invisibly.

MR. GROOVER: I'm sorry?

JUDGE MERCADER: Is that the term invisibly that you use?

MR. GROOVER: No, Your Honor. If you look at Claim 3, for instance. Sorry. We printed these with a page break in them, I see. If we look at Claim 3, we have a CPU with main memory. We have first memory management, which virtualizes the main memory, with reference to at least a

bulk -- one bulk storage. That would be the disk, for instance. And we have a graphics accelerator, which has dedicated graphics memory and a second memory management which virtualizes -- well, performs page faulting of the texture data invisibly. So, yes, you're correct, Your Honor, that that does say that the second memory management unit is operating invisibly to the CPU.

JUDGE MERCADER: So in the other claim when you referred to automatically, when you say automatically, is that your intent to say invisibly? That they're intended to be --

MR. GROOVER: I guess if I were to translate that into a longer phrase, I would say without intervention by the CPU. I think it works out to the same in this context. But nothing has to be done, nothing has to be initiated by the CPU. Whereas, you know, again, in Kaiser, which is not very close art, there are certain things that the CPU has to do. For instance, the CPU has to initiate some of the synchronization. And if this goes to -- if this does hit a page fault and call for CPU handling, as is just briefly mentioned, then the CPU will have to handle that and return or -- return control to the graphics engine.

JUDGE EASTHOM: Are you -- you're not suggesting that your -- well, I don't know, your -- I understand the part about the page fault being invisible. But you're saying any other syncing with the graphics engine in Kaiser would be something that isn't invisible to the host? Are you -- doesn't your graphics processor have some communication with your host?

MR. GROOVER: Yes, indeed. Yes, indeed. But our graphics processor, as I colorfully said, our graphics processor, if it needs something from main memory, it's able to sneak out and get it.

JUDGE EASTHOM: Right. That's the invisible part of the --

MR. GROOVER: Right. But also it can do that without any help from the host. In other words, we have -- let's look back at Claim 3. We have a first memory management and then we have a second memory management which manages texture data and performs page faulting -- I'm sorry, you were asking about the automatically one. Let's go down to Claim 4 then, please. Excuse me. Last paragraph of Claim 4, when the graphics accelerator attempts to access texture data which is in the host's physical memory, the graphics memory manager fetches that automatically, i.e., it doesn't have to be prompted by anything. This is one of the different ways to look at the implications of its having its own memory management.

And, Your Honors, I apologize for the diversity of claims in this case. I had, frankly, expected we were going to get restricted out long ago. But this is a pretty pioneering invention as best I can tell, and in those cases I try hard to find, you know, different perspectives to look at the new invention from. So that's why there's different language and different formulations in these numerous claims.

JUDGE SAADAT: Mr. Groover, your intention was to have terms invisibly and automatically mean the same thing or --

MR. GROOVER: Yes, Your Honor, I think so. At this point I'm not sure if I can say correctly what my intention was in 1999 and 2000 when I was writing this application.

JUDGE SAADAT: Let's talk about now, though.

MR. GROOVER: But, yes, I'm willing to specify now that we intend invisibly and automatically to mean the same thing. Your Honors -- sorry.

JUDGE EASTHOM: Well, thank you. It was very instructive.

JUDGE SAADAT: We -- or we think we understand.

MR. GROOVER: Okay. This is prosecution has not been a -- as far as bringing a case, let's say, to a point for clear definition of the issues, this prosecution has not been very focused in that respect. But we've tried to set forth our elements on this.

So at this time I don't need to cancel Claim 1 or you don't feel that would be --

JUDGE EASTHOM: I don't think -- I think with your argument, you focused your argument on the page faulting invisibly the host processor, and that's what Claim 1 is. I don't see how we could parcel it out from the other --

MR. GROOVER: Got it.

JUDGE EASTHOM: -- with a straight face, but --

MR. GROOVER: Okay. Your Honors, thank you very much.

JUDGE EASTHOM: Thank you.

JUDGE SAADAT: Thank you. Have a good day.

(Whereupon, the hearing concluded on August 11, 2009.)